Ex.10 Design of Content-based Recommender System

October 26, 2023

[ ]:
```
Bewin Felix R A
URK21CS1128
```

[ ]:
```
AIM:
    To Develop an E-commerce item recommender system with content-based␣
 ↪recommendation system
using the Term-Frequency Inverse Document Frequency (TF IDF) and cosine␣
 ↪similarity.
```

[ ]:
```
DESCRIPTION:

A recommendation system, often referred to as a recommender system, is a type of
software or algorithm that provides personalized suggestions to users. These
suggestions can be for products, services, content, or items of interest based␣
 ↪on
a user's past behavior, preferences, or the behavior of similar users.
Recommendation systems are widely used in various domains, including e-commerce,
content streaming, social media, and more, to enhance user experience and␣
 ↪engagement.

A search engine is a software application or online service that helps users␣
 ↪find
information on the internet or within a specific dataset. It works by indexing␣
 ↪and
cataloging vast amounts of web content and providing users with a way to search␣
 ↪for
and access relevant information quickly.

Term Frequency-Inverse Document Frequency, commonly abbreviated as TF-IDF, is a
numerical statistic used in information retrieval and text mining to evaluate
the importance of a word in a document relative to a collection of documents
(corpus). TF-IDF is a technique that combines two key components:

Term Frequency (TF):

Term Frequency measures how frequently a term (word or phrase) appears in a
document. It is calculated as the number of times a term occurs in a document
```

```
divided by the total number of terms in the document. The idea is to give higher
weight to terms that appear more frequently within a document.

Inverse Document Frequency (IDF):

Inverse Document Frequency calculates the importance of a term across a␣
↪collection
of documents. It is used to downweight common terms that appear in many␣
↪documents
and give higher weight to rare terms.

The TF-IDF score for a term in a document combines both the TF and IDF␣
↪components
to determine how important the term is in that particular document within the
context of the entire corpus. It is calculated as follows:

TF-IDF(t, d, corpus) = TF(t, d) * IDF(t, corpus)

# from sklearn.feature_extraction.text import TfidfVectorizer
# tfidf = TfidfVectorizer(stop_words='english')
# tfidf_matrix = tfidf.fit_transform(content)

Cosine similarity
Cosine similarity measures the similarity between two vectors. Since TF-IDF␣
↪returns
vectors showing the score a document gets versus the corpus, we can use cosine
similarity to identify the closest matches after we’ve used TF-IDF to generate␣
↪the
vectors.
# from sklearn.metrics.pairwise import cosine_similarity
# from sklearn.metrics.pairwise import linear_kernel
```

[ ]:
```
2. Develop an E-commerce item recommender system with content-based␣
↪recommendation
using the scikit-learn
a. Use the column: 'product'.
```

[1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import linear_kernel
```

[3]:
```
print(1128)
df = pd.read_csv("shop_details.csv")
df
```

```
1128
```

[3]:
```
        id                    product  \
0     1.0      Active classic boxers
1     2.0  Active sport boxer briefs
2     3.0        Active sport briefs
3     4.0         Alpine guide pants
4     5.0            Alpine wind jkt
…   …                        …
2182  NaN                        NaN
2183  NaN                        NaN
2184  NaN                        NaN
2185  NaN                        NaN
2186  NaN                        NaN

                                            description
0     There's a reason why our boxers are a cult fav…
1     Skinning up Glory requires enough movement wit…
2     These superbreathable no-fly briefs are the mi…
3     Skin in, climb ice, switch to rock, traverse a…
4     On high ridges, steep ice and anything alpine,…
…                                                   …
2182                                              NaN
2183                                              NaN
2184                                              NaN
2185                                              NaN
2186                                              NaN

[2187 rows x 3 columns]
```

[4]:
```python
#b. Remove the leading and trailing whitespaces in that column.
print(1128)
df["product"].str.strip()
df["product"]
```

```
1128
```

[4]:
```
0           Active classic boxers
1       Active sport boxer briefs
2             Active sport briefs
3              Alpine guide pants
4                 Alpine wind jkt
                 …
2182                          NaN
2183                          NaN
2184                          NaN
2185                          NaN
2186                          NaN
```

```
Name: product, Length: 2187, dtype: object
```

[5]:
```
#removing NaN values from the dataset.
print(1128)
df.dropna(inplace=True)
df
```

```
1128
```

[5]:
```
         id                    product  \
0       1.0       Active classic boxers
1       2.0   Active sport boxer briefs
2       3.0         Active sport briefs
3       4.0          Alpine guide pants
4       5.0             Alpine wind jkt
..      …                           …
495   496.0              Cap 2 bottoms
496   497.0                 Cap 2 crew
497   498.0              All-time shell
498   499.0       All-wear cargo shorts
499   500.0             All-wear shorts

                                            description
0    There's a reason why our boxers are a cult fav…
1    Skinning up Glory requires enough movement wit…
2    These superbreathable no-fly briefs are the mi…
3    Skin in, climb ice, switch to rock, traverse a…
4    On high ridges, steep ice and anything alpine,…
..                                                …
495  Cut loose from the maddening crowds and search…
496  This crew takes the edge off fickle weather. I…
497  No need to use that morning Times as an umbrel…
498  All-Wear Cargo Shorts bask in the glory of swe…
499  Time to simplify? Our All-Wear shorts prove th…

[500 rows x 3 columns]
```

[6]:
```
df.columns
```

[6]:
```
Index(['id', 'product', 'description'], dtype='object')
```

[7]:
```
# c. Perform feature extraction using Term Frequency Inverse Document Frequency␣
↪(TF-
#IDF).
print(1128)
indices = pd.Series(df.index, index=df['product']).drop_duplicates()
content = df['description'].fillna('')
```

```
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(content)
print("TF IDF Matrix:",tfidf_matrix)
```

```
1128
TF IDF Matrix:   (0, 2550)      0.10398386495339977
  (0, 294)      0.16071014411893011
  (0, 4478)     0.026691279623776477
  (0, 1804)     0.08733693134440922
  (0, 1057)     0.09201923404633659
  (0, 2741)     0.08577597087914687
  (0, 2665)     0.08843160800635147
  (0, 1836)     0.09471579837759175
  (0, 979)      0.06091036994464541
  (0, 1433)     0.0734929307585737
  (0, 1379)     0.0644741003780651
  (0, 703)      0.06492701570977669
  (0, 4256)     0.11673739714671233
  (0, 1569)     0.05253687697191237
  (0, 793)      0.08957389399439541
  (0, 3570)     0.10959412306927277
  (0, 2356)     0.2101475078876495
  (0, 4251)     0.05253687697191237
  (0, 1266)     0.026268438485956187
  (0, 695)      0.2101475078876495
  (0, 3052)     0.07093203632068933
  (0, 3178)     0.07093203632068933
  (0, 4077)     0.07093203632068933
  (0, 989)      0.07093203632068933
  (0, 3176)     0.07093203632068933
  :     :
  (499, 4315)   0.10777030548461138
  (499, 4099)   0.09192482716078833
  (499, 2979)   0.2243765958317256
  (499, 1391)   0.1007954104508571
  (499, 1672)   0.07744459615973451
  (499, 3690)   0.0486578839623734
  (499, 4478)   0.027391812019129633
  (499, 1804)   0.08962915376985607
  (499, 1569)   0.026957873102585114
  (499, 2356)   0.3774102234361916
  (499, 4251)   0.05391574620517023
  (499, 1266)   0.026957873102585114
  (499, 695)    0.21566298482068091
  (499, 3052)   0.07279370013042086
  (499, 3178)   0.07279370013042086
  (499, 4077)   0.07279370013042086
  (499, 989)    0.07279370013042086
```

```
  (499, 3176)	0.07279370013042086
  (499, 3595)	0.07094338003490643
  (499, 2155)	0.1392038311496402
  (499, 3)	0.0817450340489487
  (499, 2811)	0.05478362403825927
  (499, 1714)	0.07235510954312023
  (499, 1655)	0.05722767997477116
  (499, 2370)	0.05625319291837664
```

[8]:
```python
# d. Compute the cosine similarity.
print(1128)
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
print(cosine_similarities)
```

```
1128
[[1.         0.279554   0.19517104 … 0.15671646 0.18352571 0.21493076]
 [0.279554   1.         0.54659561 … 0.12053582 0.22053005 0.19737709]
 [0.19517104 0.54659561 1.         … 0.1051828  0.12856782 0.15275201]
 …
 [0.15671646 0.12053582 0.1051828  … 1.         0.11754784 0.14264239]
 [0.18352571 0.22053005 0.12856782 … 0.11754784 1.         0.57147933]
 [0.21493076 0.19737709 0.15275201 … 0.14264239 0.57147933 1.        ]]
```

[9]:
```python
# e. Display the top 'n' suggestions with the similarity score for the given␣
 ↪user input.
print(1128)
def get_recommendations(df, column, value, cosine_similarities, n):



    # Return indices for the target dataframe column and drop any duplicates
    indices = pd.Series(df.index, index=df[column]).drop_duplicates()

    # Get the index for the target value
    target_index = indices[value]

    # Get the cosine similarity scores for the target value
    cosine_similarity_scores =␣
 ↪list(enumerate(cosine_similarities[target_index]))

    # Sort the cosine similarities in order of closest similarity
    cosine_similarity_scores = sorted(cosine_similarity_scores, key=lambda x:␣
 ↪x[1], reverse=True)

    # Return tuple of the requested closest scores excluding the target item␣
 ↪and index
    cosine_similarity_scores = cosine_similarity_scores[1:n+1]

    # Extract the tuple values
```

```
    index = (x[0] for x in cosine_similarity_scores)
    scores = (x[1] for x in cosine_similarity_scores)

    # Get the indices for the closest items
    recommendation_indices = [i[0] for i in cosine_similarity_scores]

    # Get the actutal recommendations
    recommendations = df[column].iloc[recommendation_indices]

    # Return a dataframe
    df = pd.DataFrame(list(zip(index, recommendations, scores)),
                      columns=['index','recommendation',␣
 ↪'cosine_similarity_score'])

    return df

n = int(input("Enter the no.of suggetions:"))
recommendations = get_recommendations(df,
                                      'product',
                                      'Flying fish t-shirt',
                                      cosine_similarities,n)
```

```
1128
```

```
Enter the no.of suggetions: 2
```

[10]:
```
print(1128)
recommendations.head(10)
```

```
1128
```

[10]:

	index	recommendation	cosine_similarity_score
0	57	'73 logo t-shirt	0.911366
1	63	Gpiw classic t-shirt	0.892282

[ ]:
```
RESULT:
    The E-commerce item recommender system with content-based recommendation
using scikit-learn methods has been developed and the output is displayed␣
 ↪successfully.
```